Nyilvános kulcsú rejtjelezés

PPKE, ITK
Csapodi Márton

New directions in Cryptography

Whitfield Diffie & Martin E. Hellman, New directions in Cryptography, IEEE Transactions on Information Theory (1976)

I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

New directions in Cryptography

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels

Public key distribution systems *public key cryptosystem*

New directions in Cryptography

A second problem, amenable to cryptographic solution, which stands in the way of replacing contemporary business communications by teleprocessing systems is authentication. In current business, the validity of contracts is guaranteed by signatures. A signed contract serves as legal evidence of an agreement which the holder can present in court if necessary. The use of signatures, however, requires the transmission and storage of written contracts. In order to have a purely digital replacement for this paper instrument, each user must be able to produce a message whose authenticity can be checked by anyone, but which could not have been produced by anyone else, even the recipient. Since only one person can originate messages but many people can receive messages, this can be viewed as a broadcast cipher. Current electronic authentication techniques cannot meet this need.

New directions in Cryptography

III. Public Key Cryptography

We propose that it is possible to develop systems of the type shown in Fig. 2, in which two parties communicating solely over a public channel and using only publicly known techniques can create a secure connection. We examine two approaches to this problem, called public key cryptosystems and public key distribution systems, respectively. The first are more powerful, lending themselves to the solution of the authentication problems treated in the next section, while the second are much closer to realization.

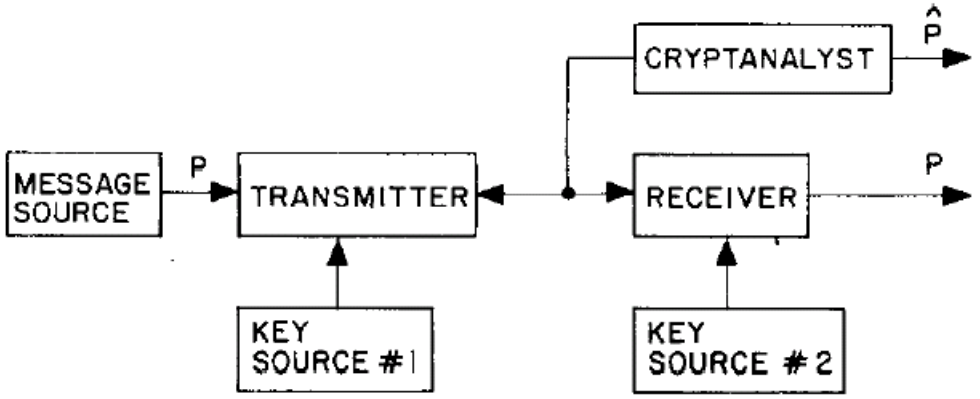

Fig. 2. Flow of information in public key system.

New directions in Cryptography

A *public key cryptosystem* is a pair of families $\{E_K\}_{K \in \{K\}}$ and $\{D_K\}_{K \in \{K\}}$ of algorithms representing invertible transformations,

$$E_K : \{M\} \rightarrow \{M\} \tag{2}$$

$$D_K : \{M\} \rightarrow \{M\} \tag{3}$$

on a finite message space $\{M\}$, such that

1) for every $K \in \{K\}$, E_K is the inverse of D_K,
2) for every $K \in \{K\}$ and $M \in \{M\}$, the algorithms E_K and D_K are easy to compute,
3) for almost every $K \in \{K\}$, each easily computed algorithm equivalent to D_K is computationally infeasible to derive from E_K,
4) for every $K \in \{K\}$, it is feasible to compute inverse pairs E_K and D_K from K.

New directions in Cryptography

A suggestive, although unfortunately useless, example of a public key cryptosystem is to encipher the plaintext, represented as a binary n-vector $\boldsymbol{m}$, by multiplying it by an invertible binary $n \times n$ matrix E. The cryptogram thus equals $E\boldsymbol{m}$. Letting $D = E^{-1}$ we have $\boldsymbol{m} = D\boldsymbol{c}$. Thus, both enciphering and deciphering require about n^2 operations. Calculation of D from E, however, involves a matrix inversion which is a harder problem. And it is at least conceptually simpler to obtain an arbitrary pair of inverse matrices than it is to invert a given matrix. Start with the identity matrix I and do elementary row and column operations to obtain an arbitrary invertible matrix E. Then starting with I do the inverses of these same elementary operations in reverse order to obtain $D = E^{-1}$. The sequence of elementary operations could be easily determined from a random bit string.

Unfortunately, matrix inversion takes only about n^3 operations. The ratio of "cryptanalytic" time (i.e., computing D from E) to enciphering or deciphering time is thus at most n, and enormous block sizes would be required to obtain ratios of 10^6 or greater. Also, it does not appear that knowledge of the elementary operations used to obtain E from I greatly reduces the time for computing D. And, since there is no round-off error in binary arithmetic, numerical stability is unimportant in the matrix inversion. In spite of its lack of practical utility, this matrix example is still useful for clarifying the relationships necessary in a public key cryptosystem.

A more practical approach to finding a pair of easily computed inverse algorithms E and D; such that D is hard to infer from E, makes use of the difficulty of analyzing programs in low level languages. Anyone who has tried to determine what operation is accomplished by someone else's machine language program knows that E itself (i.e., what E does) can be hard to infer from an algorithm for E. If the program were to be made purposefully confusing through addition of unneeded variables and statements, then determining an inverse algorithm could be made very difficult. Of course, E must be complicated enough to prevent its identification from input–output pairs.

Essentially what is required is a one-way compiler: one which takes an easily understood program written in a high level language and translates it into an incomprehensible program in some machine language. The compiler is one-way because it must be feasible to do the compilation, but infeasible to reverse the process. Since efficiency in size of program and run time are not crucial in this application, such compilers may be possible if the structure of the machine language can be optimized to assist in the confusion.

New directions in Cryptography

We now suggest a new public key distribution system

The new technique makes use of the apparent difficulty of computing logarithms over a finite field $GF(q)$ with a prime number q of elements. Let

$$Y = \alpha^X \bmod q, \qquad \text{for } 1 \leq X \leq q-1, \tag{4}$$

where α is a fixed primitive element of $GF(q)$, then X is referred to as the logarithm of Y to the base α, mod q:

$$X = \log_\alpha Y \bmod q, \qquad \text{for } 1 \leq Y \leq q-1. \tag{5}$$

Calculation of Y from X is easy, taking at most $2 \times \log_2 q$ multiplications [6, pp. 398–422]. For example, for $X = 18$,

$$Y = \alpha^{18} = (((\alpha^2)^2)^2)^2 \times \alpha^2. \tag{6}$$

Computing X from Y, on the other hand can be much more difficult and, for certain carefully chosen values of q, requires on the order of $q^{1/2}$ operations, using the best known algorithm [7, pp. 9, 575–576], [8].

New directions in Cryptography

Each user generates an independent random number X_i chosen uniformly from the set of integers $\{1,2,\cdots,q-1\}$. Each keeps X_i secret, but places

$$Y_i = \alpha^{X_i} \bmod q \tag{7}$$

in a public file with his name and address. When users i and j wish to communicate privately, they use

$$K_{ij} = \alpha^{X_i X_j} \bmod q \tag{8}$$

as their key. User i obtains K_{ij} by obtaining Y_j from the public file and letting

$$K_{ij} = Y_j^{X_i} \bmod q \tag{9}$$

$$= (\alpha^{X_j})^{X_i} \bmod q \tag{10}$$

$$= \alpha^{X_j X_i} = \alpha^{X_i X_j} \bmod q. \tag{11}$$

User j obtains K_{ij} in the similar fashion

$$K_{ij} = Y_i^{X_j} \bmod q. \tag{12}$$

Another user must compute K_{ij} from Y_i and Y_j, for example, by computing

$$K_{ij} = Y_i^{(\log_\alpha Y_j)} \bmod q. \tag{13}$$

We thus see that if logs mod q are easily computed the system can be broken. While we do not currently have a proof of the converse (i.e., that the system is secure if logs mod q are difficult to compute), neither do we see any way to compute K_{ij} from Y_i and Y_j without first obtaining either X_i or X_j.

New directions in Cryptography

IV. One-Way Authentication

The problem of authentication is perhaps an even more serious barrier to the universal adoption of telecommunications for business transactions than the problem of key distribution. Authentication is at the heart of any system involving contracts and billing. Without it, business cannot function. Current electronic authentication systems cannot meet the need for a purely digital, unforgeable, message dependent signature. They provide protection against third party forgeries, but do not protect against disputes between transmitter and receiver.

In order to develop a system capable of replacing the current written contract with some purely electronic form of communication, we must discover a digital phenomenon with the same properties as a written signature. It must be easy for anyone to recognize the signature as authentic, but impossible for anyone other than the legitimate signer to produce it. We will call any such technique *one-way authentication*. Since any digital signal can be copied precisely, a true digital signature must be recognizable without being known.

New directions in Cryptography

Consider the "login" problem in a multiuser computer system. When setting up his account, the user chooses a password which is entered into the system's password directory. Each time he logs in, the user is again asked to provide his password. By keeping this password secret from all other users, forged logins are prevented. This, however, makes it vital to preserve the security of the password directory since the information it contains would allow perfect impersonation of any user. The problem is further compounded if system operators have legitimate reasons for accessing the directory. Allowing such legitimate accesses, but preventing all others, is next to impossible.

New directions in Cryptography

This leads to the apparently impossible requirement for a new login procedure capable of judging the authenticity of passwords without actually knowing them. While appearing to be a logical impossibility, this proposal is easily satisfied. When the user first enters his password PW, the computer automatically and transparently computes a function $f(PW)$ and stores this, not PW, in the password directory. At each successive login, the computer calculates $f(X)$, where X is the proffered password, and compares $f(X)$ with the stored value $f(PW)$. If and only if they are equal, the user is accepted as being authentic. Since the function f must be calculated once per login, its computation time must be small. A million instructions (costing approximately \$0.10 at bicentennial prices) seems to be a reasonable limit on this computation. If we could ensure, however, that calculation of f^{-1} required 10^{30} or more instructions, someone who had subverted the system to obtain the password directory could not in practice obtain PW from $f(PW)$, and could thus not perform an unauthorized login. Note that $f(PW)$ is not accepted as a password by the login program since it will automatically compute $f(f(PW))$ which will not match the entry $f(PW)$ in the password directory.

New directions in Cryptography

We assume that the function f is public information, so that it is not ignorance of f which makes calculation of f^{-1} difficult. Such functions are called one-way functions and were first employed for use in login procedures by R. M. Needham [9, p. 91]. They are also discussed in two recent papers [10], [11] which suggest interesting approaches to the design of one-way functions.

More precisely, a function f is a *one-way function* if, for any argument x in the domain of f, it is easy to compute the corresponding value $f(x)$, yet, for almost all y in the range of f, it is computationally infeasible to solve the equation $y = f(x)$ for any suitable argument x.

A public key cryptosystem can be used to produce a true one-way authentication system as follows. If user A wishes to send a message M to user B, he "deciphers" it in his secret deciphering key and sends $D_A(M)$. When user B receives it, he can read it, and be assured of its authenticity by "enciphering" it with user A's public enciphering key E_A. B also saves $D_A(M)$ as proof that the message came from A. Anyone can check this claim by operating on $D_A(M)$ with the publicly known operation E_A to recover M. Since only A could have generated a message with this property, the solution to the one-way authentication problem would follow immediately from the development of public key cryptosystems.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir & L. Adleman, Communications of the ACR (1977)

Abstract

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

1. Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.
2. A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious applications in "electronic mail" and "electronic funds transfer" systems.

A message is encrypted by representing it as a number M, raising M to a publicly specified power e, and then taking the remainder when the result is divided by the publicly specified product, n, of two large secret prime numbers p and q. Decryption is similar; only a different, secret, power d is used, where $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$. The security of the system rests in part on the difficulty of factoring the published divisor, n.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

I Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem," an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

V Our Encryption and Decryption Methods

To encrypt a message M with our method, using a public encryption key (e, n), proceed as follows. (Here e and n are a pair of positive integers.)

First, represent the message as an integer between 0 and $n-1$. (Break a long message into a series of blocks, and represent each block as such an integer.) Use any standard representation. The purpose here is not to encrypt the message but only to get it into the numeric form necessary for encryption.

Then, encrypt the message by raising it to the eth power modulo n. That is, the result (the ciphertext C) is the remainder when M^e is divided by n.

To decrypt the ciphertext, raise it to another power d, again modulo n. The encryption and decryption algorithms E and D are thus:

$$\begin{aligned} C \equiv E(M) &\equiv M^e \pmod{n}, \text{ for a message } M\ . \\ D(C) &\equiv C^d \pmod{n}, \text{ for a ciphertext } C\ . \end{aligned}$$

Note that encryption does not increase the size of a message; both the message and the ciphertext are integers in the range 0 to $n-1$.

The *encryption key* is thus the pair of positive integers (e, n). Similarly, the *decryption* key is the pair of positive integers (d, n). Each user makes his encryption key public, and keeps the corresponding decryption key private. (These integers should properly be subscripted as in n_A, e_A, and d_A, since each user has his own set. However, we will only consider a typical set, and will omit the subscripts.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

How should you choose your encryption and decryption keys, if you want to use our method?

You first compute n as the product of two primes p and q:

$$n = p \cdot q \ .$$

These primes are very large, "random" primes. Although you will make n public, the factors p and q will be effectively hidden from everyone else due to the enormous difficulty of factoring n. This also hides the way d can be derived from e.

You then pick the integer d to be a large, random integer which is relatively prime to $(p-1) \cdot (q-1)$. That is, check that d satisfies:

$$\gcd(d, (p-1) \cdot (q-1)) = 1$$

The integer e is finally computed from p, q, and d to be the "multiplicative inverse" of d, modulo $(p-1) \cdot (q-1)$. Thus we have

$$e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}.$$

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

We demonstrate the correctness of the deciphering algorithm using an identity due to Euler and Fermat [7]: for any integer (message) M which is relatively prime to n,

$$M^{\phi(n)} \equiv 1 \pmod{n} . \tag{3}$$

Here $\phi(n)$ is the Euler totient function giving number of positive integers less than n which are relatively prime to n. For prime numbers p,

$$\phi(p) = p - 1 .$$

In our case, we have by elementary properties of the totient function [7]:

$$\begin{aligned} \phi(n) &= \phi(p) \cdot \phi(q) \\ &= (p-1) \cdot (q-1) \\ &= n - (p+q) + 1 . \end{aligned} \tag{4}$$

Since d is relatively prime to $\phi(n)$, it has a multiplicative inverse e in the ring of integers modulo $\phi(n)$:

$$e \cdot d \equiv 1 \pmod{\phi(n)}. \tag{5}$$

$$M^{e \cdot d} \equiv M^{k \cdot \phi(n)+1} \equiv M \pmod{n}.$$

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

B How to Find Large Prime Numbers

Each user must (privately) choose two large random numbers p and q to create his own encryption and decryption keys. These numbers must be large so that it is not computationally feasible for anyone to factor $n = p \cdot q$. (Remember that n, but not p or q, will be in the public file.) We recommend using 100-digit (decimal) prime numbers p and q, so that n has 200 digits.

To find a 100-digit "random" prime number, generate (odd) 100-digit random numbers until a prime number is found. By the prime number theorem [7], about $(\ln 10^{100})/2 = 115$ numbers will be tested before a prime is found.

To gain additional protection against sophisticated factoring algorithms, p and q should differ in length by a few digits, both $(p-1)$ and $(q-1)$ should contain large prime factors, and $\gcd(p-1, q-1)$ should be small. The latter condition is easily checked.

To find a prime number p such that $(p-1)$ has a large prime factor, generate a large random prime number u, then let p be the first prime in the sequence $i \cdot u + 1$, for $i = 2, 4, 6, \ldots$. (This shouldn't take too long.) Additional security is provided by ensuring that $(u-1)$ also has a large prime factor.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

C How to Choose d

It is very easy to choose a number d which is relatively prime to $\phi(n)$. For example, any prime number greater than $\max(p, q)$ will do. It is important that d should be chosen from a large enough set so that a cryptanalyst cannot find it by direct search.

D How to Compute e from d and $\phi(n)$

To compute e, use the following variation of Euclid's algorithm for computing the greatest common divisor of $\phi(n)$ and d. (See exercise 4.5.2.15 in [3].) Calculate $\gcd(\phi(n), d)$ by computing a series $x_0, x_1, x_2, \ldots$, where $x_0 \equiv \phi(n), x_1 = d$, and $x_{i+1} \equiv x_{i-1} \pmod{x_i}$, until an x_k equal to 0 is found. Then $\gcd(x_0, x_1) = x_{k-1}$. Compute for each x_i numbers a_i and b_i such that $x_i = a_i \cdot x_0 + b_i \cdot x_1$. If $x_{k-1} = 1$ then b_{k-1} is the multiplicative inverse of $x_1 \pmod{x_0}$. Since k will be less than $2\log_2(n)$, this computation is very rapid.

If e turns out to be less than $\log_2(n)$, start over by choosing another value of d. This guarantees that every encrypted message (except $M = 0$ or $M = 1$) undergoes some "wrap-around" (reduction modulo n) .

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

VIII A Small Example

Consider the case $p = 47, q = 59, n = p \cdot q = 47 \cdot 59 = 2773$, and $d = 157$. Then $\phi(2773) = 46 \cdot 58 = 2668$, and e can be computed as follows:

$x_0 = 2668, \quad a_0 = 1, \quad b_0 = 0,$
$x_1 = 157, \quad a_1 = 0, \quad b_1 = 1,$
$x_2 = 156, \quad a_2 = 1, \quad b_2 = -16$ (since $2668 = 157 \cdot 16 + 156$) ,
$x_3 = 1, \quad a_3 = -1, \quad b_3 = 17$ (since $157 = 1 \cdot 156 + 1$) .

Therefore $e = 17$, the multiplicative inverse $\pmod{2668}$ of $d = 157$.

With $n = 2773$ we can encode two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02, ..., Z = 26. Thus the message

```
ITS ALL GREEK TO ME
```

(Julius Caesar, I, ii, 288, paraphrased) is encoded:

```
0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
```

Since $e = 10001$ in binary, the first block ($M = 920$) is enciphered:

$$M^{17} = (((((1)^2 \cdot M)^2)^2)^2)^2 \cdot M = 948 \pmod{2773} .$$

The whole message is enciphered as:

```
0948 2342 1084 1444 2663 2390 0778 0774 0219 1655 .
```

The reader can check that deciphering works: $948^{157} \equiv 920 \pmod{2773}$, etc.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

We show in the next sections that all the obvious approaches for breaking our system are at least as difficult as factoring n. While factoring large numbers is not provably difficult, it is a well-known problem that has been worked on for the last three hundred years by many famous mathematicians. Fermat (1601?-1665) and Legendre (1752-1833) developed factoring algorithms; some of today's more efficient algorithms are based on the work of Legendre. As we shall see in the next section, however, no one has yet found an algorithm which can factor a 200-digit number in a reasonable amount of time. We conclude that our system has already been partially "certified" by these previous efforts to find efficient factoring algorithms.

In the following sections we consider ways a cryptanalyst might try to determine the secret decryption key from the publicly revealed encryption key. We do not consider ways of protecting the decryption key from theft; the usual physical security methods should suffice. (For example, the encryption device could be a separate device which could also be used to generate the encryption and decryption keys, such that the decryption key is never printed out (even for its owner) but only used to decrypt messages. The device could erase the decryption key if it was tampered with.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

A Factoring n

Factoring n would enable an enemy cryptanalyst to "break" our method. The factors of n enable him to compute $\phi(n)$ and thus d. Fortunately, factoring a number seems to be much more difficult than determining whether it is prime or composite.

Table 1 gives the number of operations needed to factor n with Schroeppel's method, and the time required if each operation uses one microsecond, for various lengths of the number n (in decimal digits).

Table 1

Digits	*Number of operations*	*Time*
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

B Computing $\phi(n)$ Without Factoring n

If a cryptanalyst could compute $\phi(n)$ then he could break the system by computing d as the multiplicative inverse of e modulo $\phi(n)$ (using the procedure of Section VII D).

We argue that this approach is no easier than factoring n since it enables the cryptanalyst to easily factor n using $\phi(n)$. This approach to factoring n has not turned out to be practical.

How can n be factored using $\phi(n)$? First, $(p+q)$ is obtained from n and $\phi(n) = n-(p+q)+1$. Then $(p-q)$ is the square root of $(p+q)^2-4n$. Finally, q is half the difference of $(p+q)$ and $(p-q)$.

Therefore breaking our system by computing $\phi(n)$ is no easier than breaking our system by factoring n. (This is why n must be composite; $\phi(n)$ is trivial to compute if n is prime.)

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

C Determining d Without Factoring n or Computing $\phi(n)$.

Of course, d should be chosen from a large enough set so that a direct search for it is unfeasible.

We argue that computing d is no easier for a cryptanalyst than factoring n, since once d is known n could be factored easily. This approach to factoring has also not turned out to be fruitful.

A knowledge of d enables n to be factored as follows. Once a cryptanalyst knows d he can calculate $e \cdot d - 1$, which is a multiple of $\phi(n)$. Miller [6] has shown that n can be factored using any multiple of $\phi(n)$. Therefore if n is large a cryptanalyst should not be able to determine d any easier than he can factor n.

A cryptanalyst may hope to find a d' which is equivalent to the d secretly held by a user of the public-key cryptosystem. If such values d' were common then a brute-force search could break the system. However, all such d' differ by the least common multiple of $(p-1)$ and $(q-1)$, and finding one enables n to be factored. (In (3) and (5), $\phi(n)$ can be replaced by $\text{lcm}(p-1, q-1)$.) Finding any such d' is therefore as difficult as factoring n.

A method for Obtaining Digital Signatures and Public-Key Cryptosystems

D Computing D in Some Other Way

Although this problem of "computing e-th roots modulo n without factoring n" is not a well-known difficult problem like factoring, we feel reasonably confident that it is computationally intractable. It may be possible to prove that any general method of breaking our scheme yields an efficient factoring algorithm. This would establish that any way of breaking our scheme must be as difficult as factoring. We have not been able to prove this conjecture, however.

Our method should be certified by having the above conjecture of intractability withstand a concerted attempt to disprove it. The reader is challenged to find a way to "break" our method.